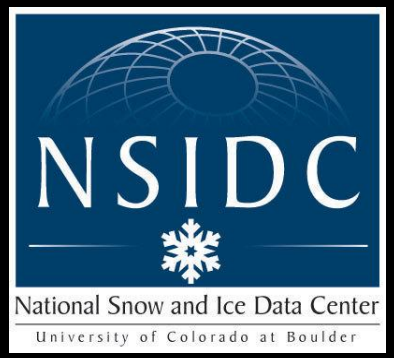
Strategies, motivations, and influencing adoption of testing for scientific code



Ian Truslove, Erik Jasiak

National Snow and Ice Data Center
University of Colorado Boulder

BESSIG, 18 September 2013

Bios and Intros

Ian Truslove

Software Engineer
National Snow and Ice Data Center
@iantruslove
ian.truslove@nsidc.org

Erik Jasiak

Head of Software Development / Portfolio Manager
National Snow and Ice Data Center
@erikjasiak
erik.jasiak@nsidc.org

DOI: 10.7265/N5SF2T4S

Aims

Make a case for using tests to improve valuable functional and non-functional attributes of scientific and research code.

Sketch out part of the spectrum of useful unit test related techniques.

Provide some social and organizational strategies to influence adoption.

Science Programming

Science Programming: A Second Class Citizen?

> *[...] recent studies have found that scientists typically spend 30% or more of their time developing software. However, 90% or more of them are primarily self-taught, and therefore lack exposure to basic software development practices [...]*

"Best Practices for Scientific Computing", Wilson et al, 2012
arXiv:1210.0530 [cs.MS]

Science Programming = Science

> *An article about computational science in a scientific publication isn't the scholarship itself, it's merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

- Jon Claerbout

"The Legal Framework for Reproducible Scientific Research", Stodden, 2009

Science Programming

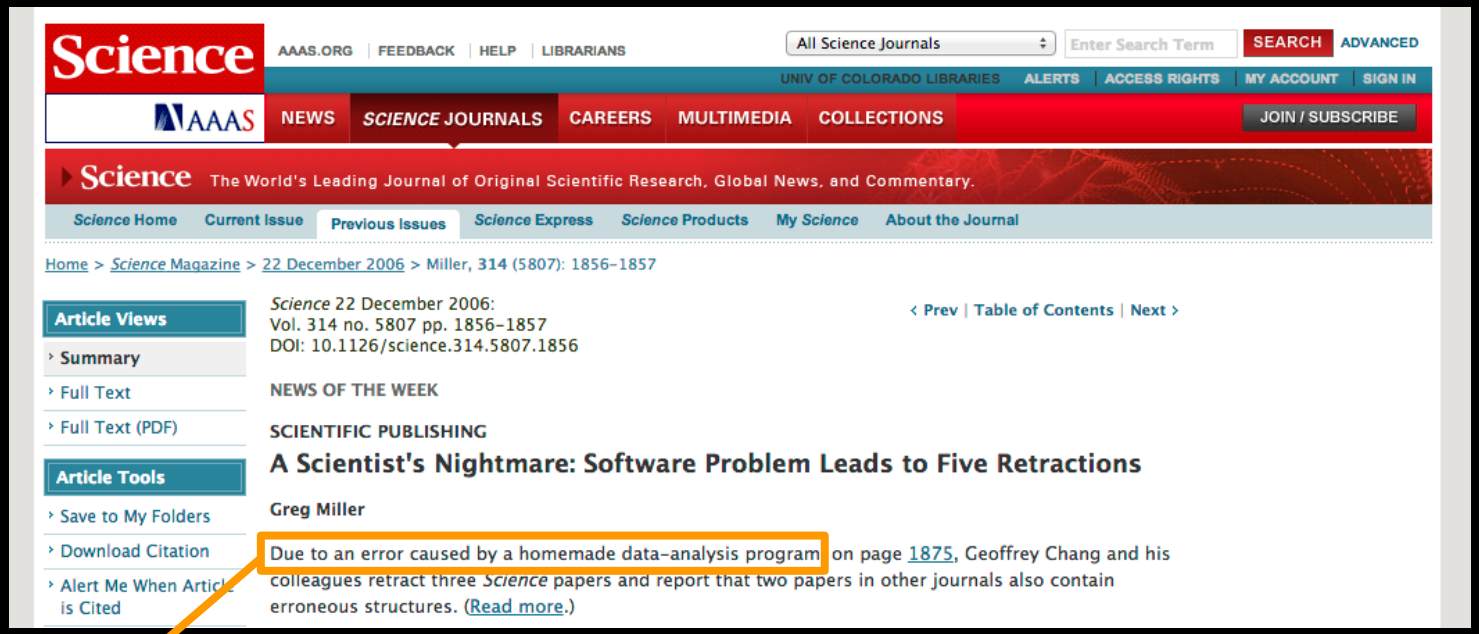
Problems:
reusability,
repeatability

Big Problems:
errors, retractions

Challenge:
Open Science

http://sidoxia.files.wordpress.com/2012/07/coin-side.jpg

Your Nightmare?



Science AAAS.ORG | FEEDBACK | HELP | LIBRARIANS

All Science Journals | Enter Search Term | SEARCH | ADVANCED

UNIV OF COLORADO LIBRARIES | ALERTS | ACCESS RIGHTS | MY ACCOUNT | SIGN IN

AAAS | NEWS | SCIENCE JOURNALS | CAREERS | MULTIMEDIA | COLLECTIONS | JOIN / SUBSCRIBE

Science The World's Leading Journal of Original Scientific Research, Global News, and Commentary.

Science Home | Current Issue | Previous Issues | Science Express | Science Products | My Science | About the Journal

Home > Science Magazine > 22 December 2006 > Miller, 314 (5807): 1856–1857

Article Views
- Summary
- Full Text
- Full Text (PDF)

Article Tools
- Save to My Folders
- Download Citation
- Alert Me When Article is Cited

Science 22 December 2006:
Vol. 314 no. 5807 pp. 1856–1857
DOI: 10.1126/science.314.5807.1856

‹ Prev | Table of Contents | Next ›

NEWS OF THE WEEK

SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Greg Miller

Due to an error caused by a homemade data-analysis program on page 1875, Geoffrey Chang and his colleagues retract three *Science* papers and report that two papers in other journals also contain erroneous structures. (Read more.)

"Due to an error caused by a *homemade* data-analysis program […]"

http://www.sciencemag.org/content/314/5807/1856

Towards Scientific Software Engineering

Software Engineering

https://twitter.com/fogus/status/379394655500394496

“Best Practices for Scientific Computing”

1. Write programs for people, not computers.
2. Automate repetitive tasks.
3. Use the computer to record history.
4. Make incremental changes.
5. Use version control.
6. Don’t repeat yourself (or others).
7. Plan for mistakes.
8. Optimize software only after it works correctly.
9. Document design and purpose, not mechanics.
10. Collaborate.

“Best Practices for Scientific Computing”, Wilson et al, 2012
arXiv:1210.0530 [cs.MS]

Proposed path for improvement: automated tests

Tests are executable specifications: repeatable, independent verification that the code is correct.

Huge spectrum of techniques: unit tests, test driven development, acceptance tests...

Not controversial: Beck, Feathers, Fowler, Hunt & Thomas (PragProg), Martin, Subramaniam, ...

Testing What?

Unit tests primarily concerned with internal quality.

Integration tests, Acceptance tests concerned with external quality.

“Growing Object-Oriented Software, Guided by Tests”, Freeman & Pryce, 2010

About Unit Tests

Small piece of code to test another piece of code.

Expose the behavior of a particular *unit* of code (e.g. a method or function).

Typically written with a framework providing structural and assertion constructs.

Highlights regressions from the contract.

Good Unit Tests

FIRST:

- Fast
- Independent
- Repeatable
- Self-verifying
- Timely

More on Good Unit Tests

Embody good software design

...but with an emphasis on readability.

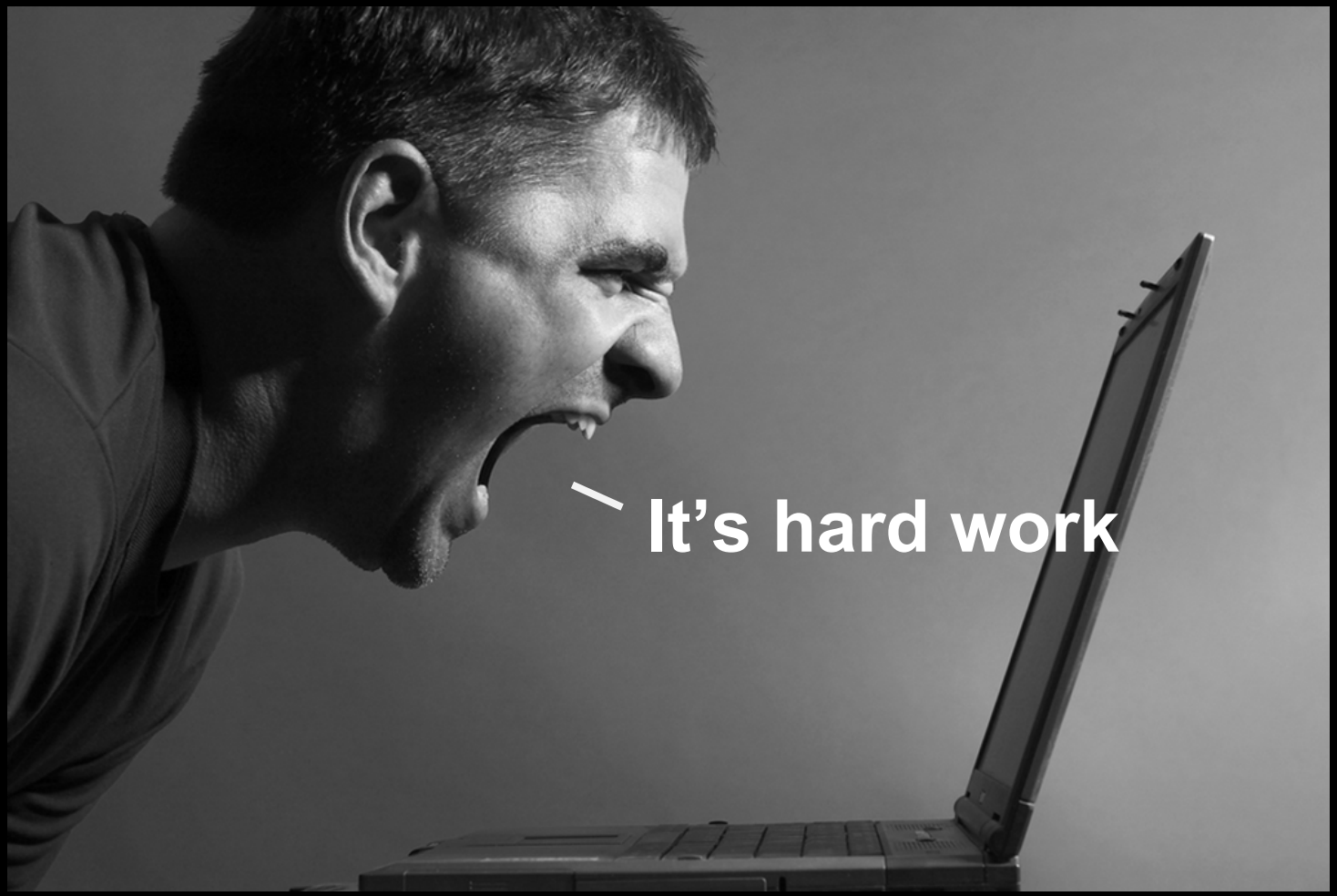
http://codinghorror.typepad.com/.a/6a0120a85dcdae970b0128776fbe0b970c-pi

Legacy Code

Legacy code = code without unit tests

...but how to get unit tests in?

Un-Legacy Code

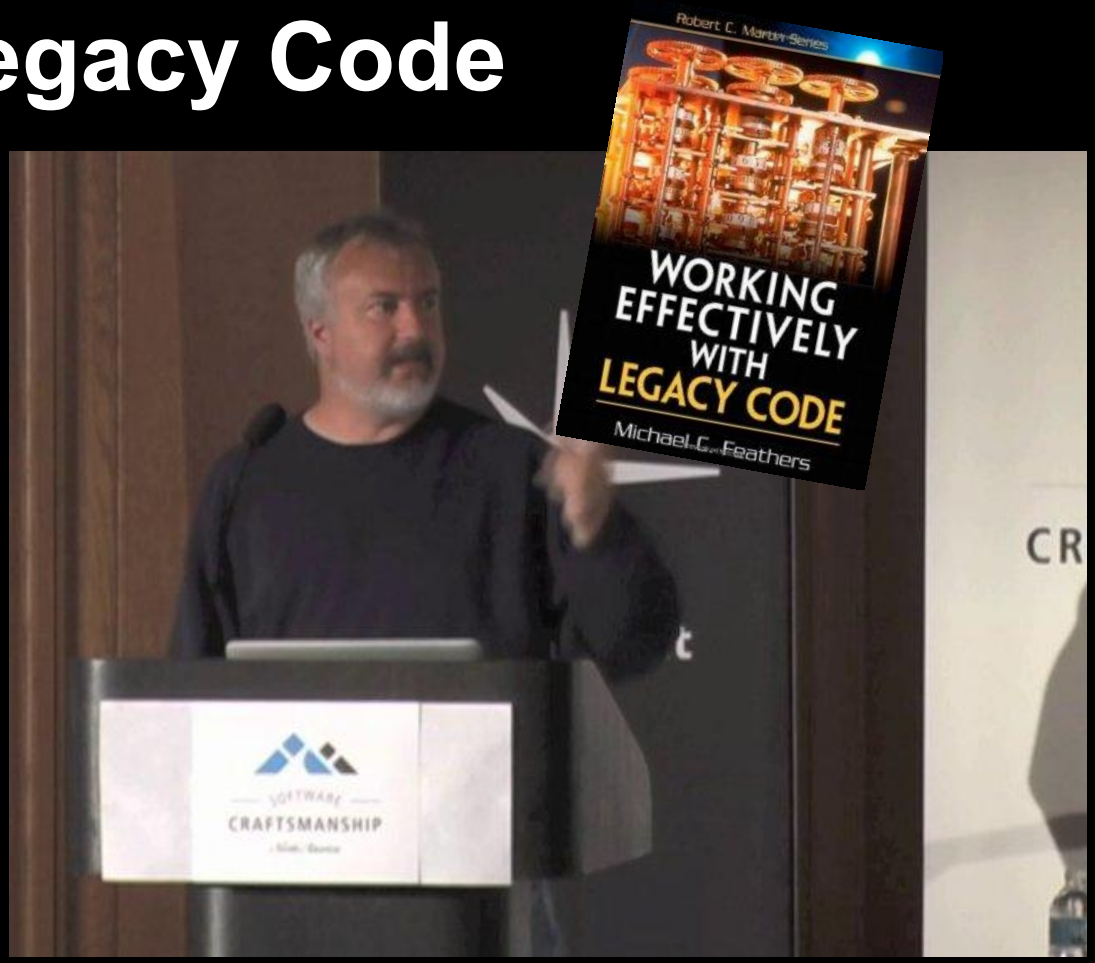


http://www.menspsychology.com/sites/default/files/facebook-frustration_full.png

Un-Legacy Code

http://worldonline.media.clients.ellingtoncms.com/img/croppedphotos/2011/04/18/trash_pick_up_day_t640.png?a6ea3ebd4438a44b86d2e9c39ecf7613005fe067

Un-Legacy Code



Feathers has great strategies

http://b.vimeocdn.com/ts/374/665/374665585_640.jpg

http://ecx.images-amazon.com/images/I/51S2-5n%2BJCL._SY300_.jpg

The Extended Test Family

Tests that aren’t unit tests:

- cover different scopes
- are still valuable
- shouldn’t be conflated with your unit tests!

Also consider stress tests, penetration tests, UI tests, ...

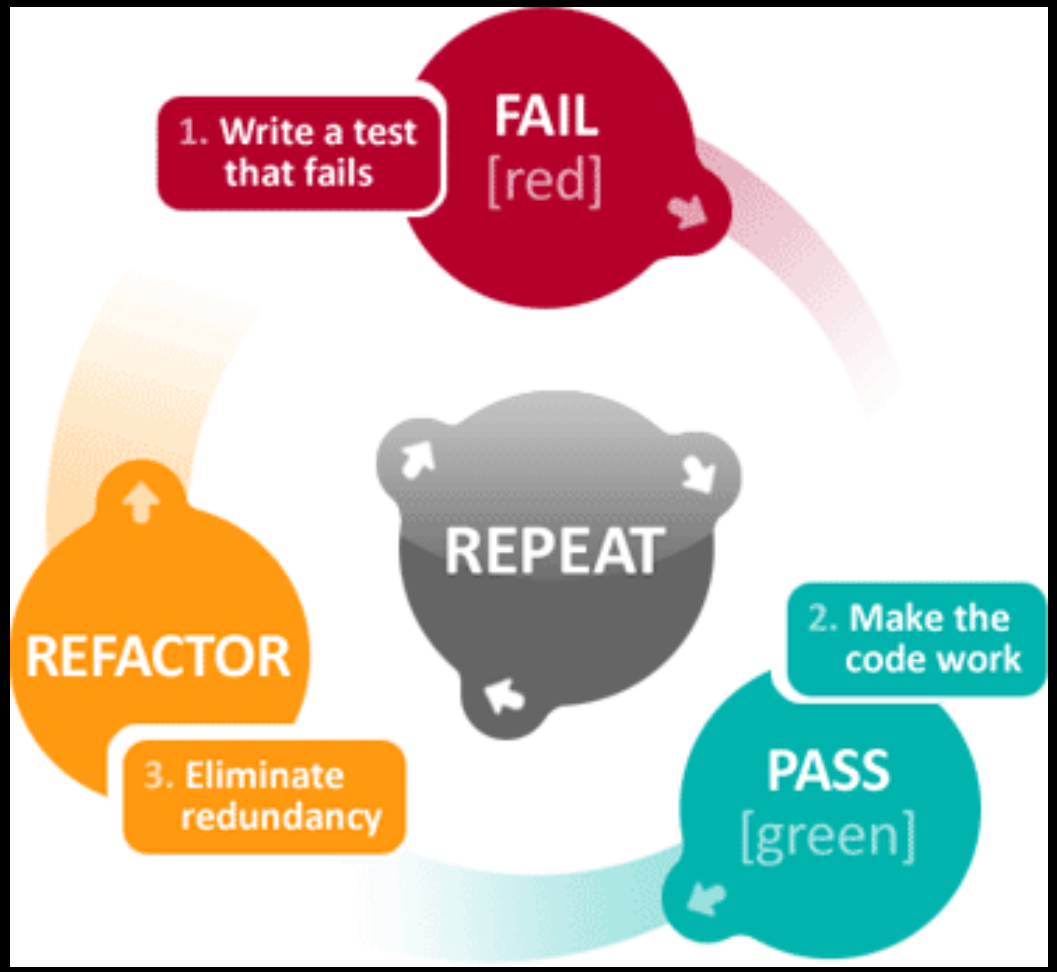
TDD - Test Driven Development

Write the tests for your code
***before** writing the code itself*

TDD is a design exercise.

http://farm4.static.flickr.com/3087/2764892330_a8ffb86dcf.jpg

About TDD



http://www.agileapps.co.uk/images/methodology/tdd-red-green-refactor-diagram.gif

About TDD

- Clean code
  - e.g. Inversion of Control
- Readable code
  - e.g. emphasis on names
- Lots of tests
  - good test coverage, bad if it’s not good code
- Mock objects
  - faked collaborators controlled by the tests

BDD - Behavior Driven Design

a "ubiquitous language" for analysis

Acceptance criteria should be executable

"Introducing BDD", Dan North

BDD: A language of executable specifications

```
1  require 'spec_helper'
2
3  describe 'Dataset Catalog Service App' do
4
5    it 'should provide dataset ISO xml at the dataset endpoint' do
6      get '/TEST_ID ', {}, { 'HTTP_ACCEPT' => 'application/x-iso19115+xml' }
7      last_response.should be_ok
8      last_response.header['Content-Type'].should match '^application/x\-iso19115\+xml'
9      last_response.body.should eql iso
10   end
11
12   it 'should provide an oai iso feed at the oai endpoint' do
13     get '/oai', { 'verb' => 'ListRecords', 'metadata_prefix' => 'iso' }
14     last_response.should be_ok
15     last_response.header['Content-Type'].should match '^application/xml'
16     last_response.body.should include(iso)
17   end
18 end
```

Acceptance Tests

End-to-end tests that verify the code *en bloc* meets the (user-facing) requirements.

Acceptance Tests: A domain-specific language for testing

```
Feature: We can apply a 3x3 neighborhood minimum filter to the data

  Scenario: Multiple minimums
    Given we have the following data layer:
      |   |   0 |   1 |   2 |   3 |   4 |
      | 0 | 100 | 100 | 100 | 100 | 100 |
      | 1 | 100 |  40 | 100 | 100 | 100 |
      | 2 | 100 | 100 |  50 | 100 | 100 |
      | 3 | 100 | 100 | 100 | 100 | 100 |
      | 4 | 100 | 100 | 100 | 100 | 100 |
    When we apply a neighborhood minimum filter
    Then the data should be:
      |   |   0 |   1 |   2 |   3 |   4 |
      | 0 |  40 |  40 |  40 | 100 | 100 |
      | 1 |  40 |  40 |  40 |  50 | 100 |
      | 2 |  40 |  40 |  40 |  50 | 100 |
      | 3 | 100 |  50 |  50 |  50 | 100 |
      | 4 | 100 | 100 | 100 | 100 | 100 |
```

Acceptance Tests: another DSL for a different domain

```
Feature: Basic ISO serialization of a single data set
  Background:
    Given there are the following valid environments:
      | Environment | Hostname  | Port  | RelativeRoot   |
      | development | localhost | 1580  |                |
      | production  | frozen    | 11580 | /api/dataset/2 |

  Scenario: Successful response
    When I request the ISO document for NSIDC-0051
    Then I get a valid response
    And the response has valid content

  Scenario: Cache populated and used
    When I clear the cache for NSIDC-0051
    And I request the ISO document for NSIDC-0051
    And I request the ISO document for NSIDC-0051
    Then the second response is faster then the first response
```

The Human Side

Cultural and Management Issues

Testing argument #427

http://dilbert.com/dyn/str_strip/000000000/00000000/0000000/000000/60000/5000/600/65675/65675.strip.zoom.gif (with edits)

“The Farmer and the Engineer”

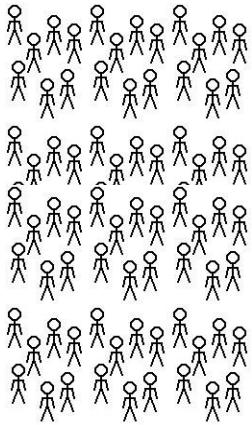
http://upload.wikimedia.org/wikipedia/commons/e/e7/Farmer_plowing_in_Fahrenwalde,_Mecklenburg-Vorpommern,_Germany.jpg

http://us.123rf.com/400wm/400/400/luislouro/luislouro0905/luislouro090500255/4824342-young-engineer-with-laptop-siting-in-a-tractor-isolated-against-a-white-background.jpg

We are all matrixed

Science Goals

Development goals

	 	 	
One Person	Two People	Team	Organization

* image citations at end

The Arguments Against ~~Testing~~ Anything

"This is going to take longer!"

- Nefarious counterpart: "Time writing X lines of code" as a measurement.

"I'm opposed to process mandates."

"Nobody cares about my code but me..."

- "...and I'm never going to show anyone anyway."

Complexity in Scientific Software

$$\mathcal{F}f[m] = \mathcal{F}f_{\text{even}}[m] + \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

$$\mathcal{F}f[m + n/2] = \mathcal{F}f_{\text{even}}[m] - \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

```
def fft(signal):
   n = len(signal)
   if n == 1:
      return signal
   else:
      Feven = fft([signal[i] for i in xrange(0, n, 2)])
      Fodd = fft([signal[i] for i in xrange(1, n, 2)])

      combined = [0] * n
      for m in xrange(n/2):
         combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
         combined[m + n/2] = Feven[m] - omega(n, -m) * Fodd[m]

      return combined
```

http://jeremykun.com/2012/07/18/the-fast-fourier-transform/

The myth of lines of code

- Award winning “best lines of code ever”
- Runs in perl **and** postscript!

The myth of lines of code

- Award winning “best lines of code ever”
- Runs in perl **and** postscript!

```
/;{}def/#{def}def/$_={/Times-Bold exch selectfont}#/_{rmoveto}#/"{dup}#/*/!/$
;/q{exch}#/x ; {/J q #}#/.{/T q #}#{stringwidth}#{}#{}# 14 string dup dup dup
260 40 moveto 90 rotate ; %/}};$0='"\e[7m \e[0m"';@ARGV=split//,reverse
q(ThePerl). q(Journal) x 220 ; q ; 0 T putinterval exch 7 J putinterval ;
 ; $_= q /m$ pop T($*!$"=!$ " )pop " * true% ? $ " $!" "  !!  !! % !" !"    !
! charpath {!"""}pop $ pop{""!}pop ! neg{!#}pop 220 ! neg _{!!}pop J false %T
charpath  clip " pop 0 " moveto 6{!!}pop $_= 105{!!}pop {$ ! $ " !  #! ##}
pop{dup dup $ ! " pop pop q{"}pop 22{dup show}repeat {"}pop q 22 mul{$ "} pop
neg{!#! $ "}pop ! 8 .65 mul{$ # # $}pop ! neg{"}pop   _ pop{"}pop } repeat pop
" {  $ " ! ! ! $ " ! !" "#"  #"!"""""! #" " # "m/;@ARGV=(@ARGV[-14..-1])x50;q}
 0 "%};s/m[ou]|[-\dA-In-z.\n_{}]|\$_=//gx;s/(.)(?{$*="})/('$*.='.(++$#
%2?":"$0;").'pop;')x(ord($1)-31).'$*'/gee;s/((.(\e\[.m)*|.){77})/$1\n/g;print
; sub showpage {}
```

The myth of lines of code - pt 2

Development effort is <u>not</u> measured in keystrokes.

How much brainpower does my code consume for everyone involved?

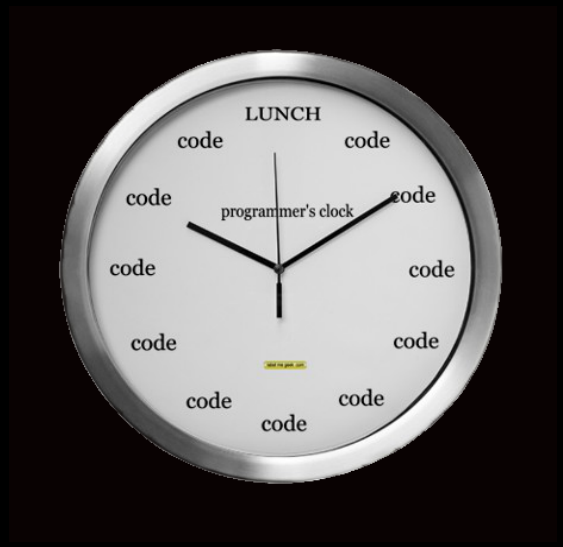
```
if (string.IsNullOrEmpty(filetext) || !filetext.Contains(",")) goto end;

*++b ? (*++b + *(b-1)) 0

getElements( ‘p’, {'class':'statusbar'} )[0].firstChild.innerText
```

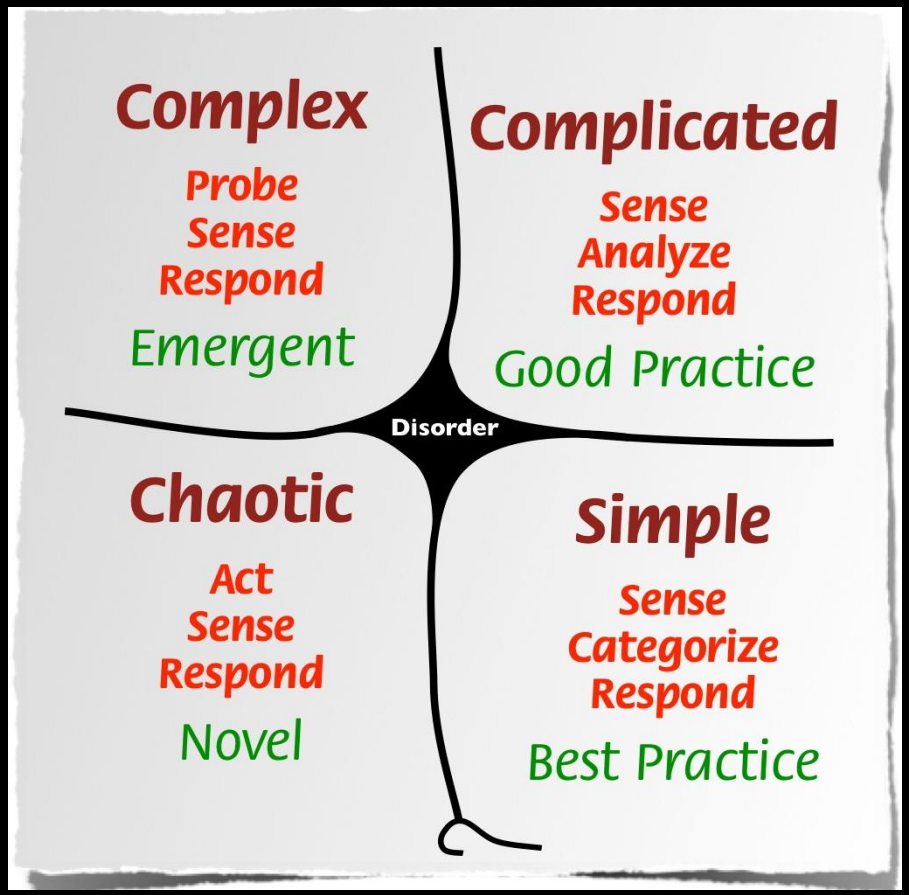
The reality of lines of code

The real point: how much time does my code take to create <u>and</u> debug <u>and</u> maintain?

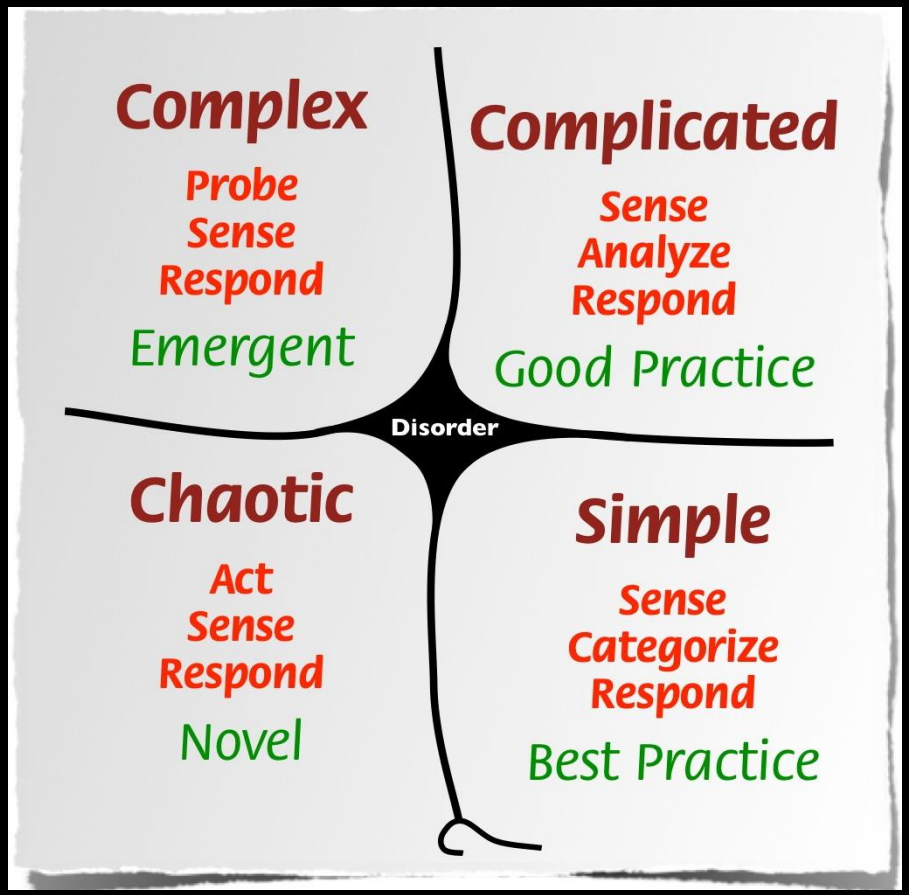


http://www.photoshop3d.com/wp-content/uploads/2007/02/hourglass_00.jpg
http://i1.cpcache.com/product/382140623/programmers_modern_wall_clock.jpg?color=Silver&height=460&width=460&qv=90
http://cdn.homedit.com/wp-content/uploads/2011/02/hwc1.jpg

Process vs good practice



Process vs good practice

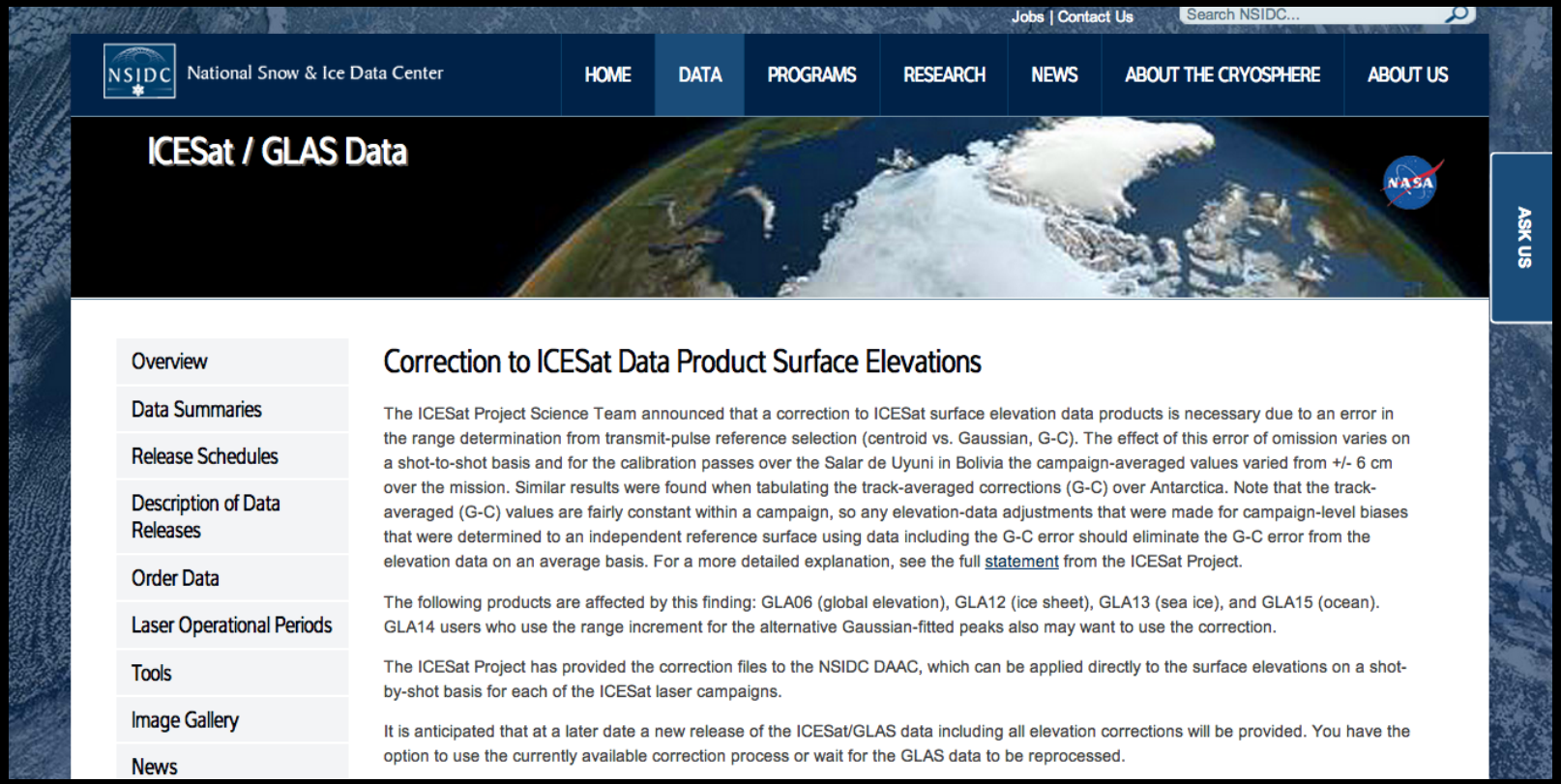


Software Engineering

Operational Procedure (Process)

http://en.wikipedia.org/wiki/File:Cynefin_framework_Feb_2011.jpeg

My code: mine alone.



My code: mine alone?

> *Data is a precious thing and will last longer than the systems themselves.*
>
> *Tim Berners-Lee*

- Who owns the results?
- How long will this be around?
- Will your work ever need to be verified?
- Will the data you produce ever need to be expanded upon?

Nuts and bolts technique:

Stroke their ego.

http://images4.wikia.nocookie.net/__cb20130418031554/poohadventures/images/5/5e/Anton_Ego.jpg

Summary: Strategies

- More lines of code: change the argument.
  - It’s about time, maintenance, efficiency
- Opposition to process / mandate: change the argument again.
  - We’re looking to mitigate complexity with good practice, not force another barrier.
- It’s my code; nobody else cares.
  - Carrot: Naked, unbridled flattery of their work.
  - (Stick: call from Bill O’Reilly if errors are found)

What you can do to help

(Hint: don’t do this.)

http://robwellsblog.files.wordpress.com/2013/08/sci-behind-testing.jpg?w=696

What you can do to help

(Hint: or this.)

http://blog.decayingcode.com/image.axd?picture=mycodecantfail_thumb.jpg

Summary of what we do at NSIDC

BDD and TDD in Python, Ruby, JavaScript, Java.

Acceptance testing, including web interfaces.

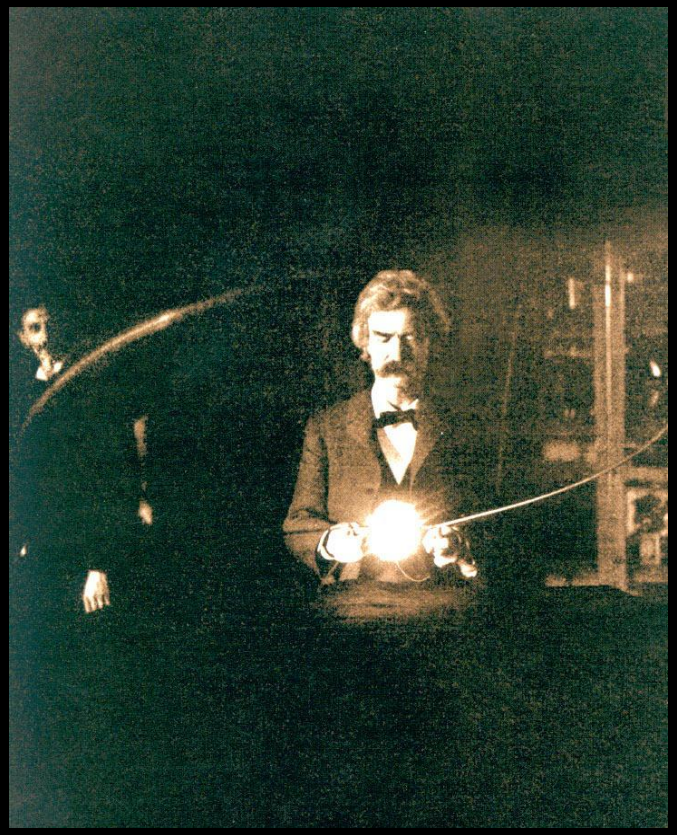
Continuous integration - all tests are run on every check in.

Continuous delivery: push-button releases.

Thank you!

@iantruslove
ian.truslove@nsidc.org

@erikjasiak
erik.jasiak@nsidc.org



Slides: http://dx.doi.org/10.7265/N5SF2T4S

http://www.teslasociety.com/pictures/ntesla4.jpg

Thank you! part 2

Additional citations: matrix org slide

www.simtalk.com/news-2-you/Penguin/Page10.htm

http://images5.cafepress.com/image/31972305_150x150.png

http://uwcpsl.files.wordpress.com/2013/02/teamwork.png

http://morganbeebe.files.wordpress.com/2011/07/stick-figure-group-of-63.jpg

The Art of Controversy: *“Theoretisch ja, praktisch nein”*

“Well, that sounds like a nice idea, but the reality with *our* work is...”

It’s just code. The point is to ensure that your ideas are correctly expressed in the code. Technical obstacles are merely speed bumps to overcome.

http://coolhaus.de/art-of-controversy/erist33.htm

The Art of Controversy: *“Das Prinzip der Schublade”*

“Testing? That’s something QA departments should do.”

At the end of the day, I’m responsible for the quality and correctness of my own work.

http://coolhaus.de/art-of-controversy/erist32.htm

The Art of Controversy: *“Ein einziges Gegenbeispiel genügt”*

“My friends Alice and Bob over at FGA did TDD, and it took them forever and halved their productivity and they had to maintain all the tests too. TDD’s not going to help.”

Of course there’s a counter-example. It isn’t necessarily easy, and you can do it wrong. Study good UTs. Gains are still gains.

http://coolhaus.de/art-of-controversy/erist25.htm

The Art of Controversy: *“Ein einziges Gegenbeispiel genügt”*

“My friends Alice and Bob over at FGA did TDD, and it took them forever and halved their productivity and they had to maintain all the tests too. TDD’s not going to help.”

Of course there’s a counter-example. It isn’t necessarily easy, and you can do it wrong. Study good UTs. Gains are still gains.

http://coolhaus.de/art-of-controversy/erist25.htm